昨日回顾

外键:

foreign key(从表的字段) references 主表(主表的主键) [选项];

选项分为 on update|delete [set null| restrict|cascade];

删除外键: 第一步: show create table 获取外键名字;

第二步: alter table 表名 drop foreign key 外键名字;

添加外键: alter table表名 add foreing key (从表字段) references 主表(主表的主键) [选项];

外键注意事项: 表的存储引擎必须为 innodb

插入数据:

蠕虫复制: insert into 表名 select 字段列表 from 表名;

插入数据的时候主键冲突:

第一种方式: replace into 表名 values 值列表

第二种方式: insert into 表名 values 值列表

on duplicate key update 字段 = 字段值

删除数据:

删除数据: delete from 表名 条件;

删除所有数据恢复主键值: truncate 表名;

删除表: drop

五子查询:

select \*|字段列表　from 表名

where 子句

group by

having

order by

limit;

where子句中 一般和 运算符,between and, in ,not in, and,or,not,like(模糊查询) %任何字符, \_ 一个字符

group by: 一般和 聚合函数配合使用 avg 平均,sum 和, count个数 max,min

having: 咋用? 意思和where一样,就是对数据进行筛选.

当没有groupy只要条件帅选的时候 having和where一样.

where后边不能跟 聚合函数

having后边可以跟聚合函数

order by: 排序, 字段按照升序(asc) 或者是降序(desc)的顺序排序

limit: 参数有一个的,表示返回几个

参数2个的,第一个参数表示: 从哪里开始,第二个参数:每页显示几个

$page = 1

$numofperpage = 20;

limit ($page-1)\*$numofperpage,$numofperpage;

# 第23讲 联合查询

## 联合查询介绍

数据表数据量比较大时，存储时按照一定的方式分割数据表，将数据存放在不同的数据表中。可以使用union查询满足一定条件的全部信息

## 语法

select 子句

union [选项]

select 子句

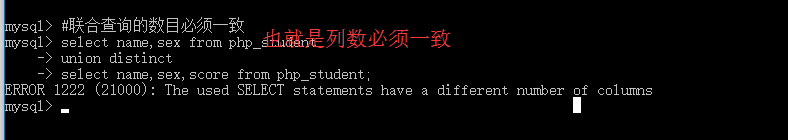
union [选项]

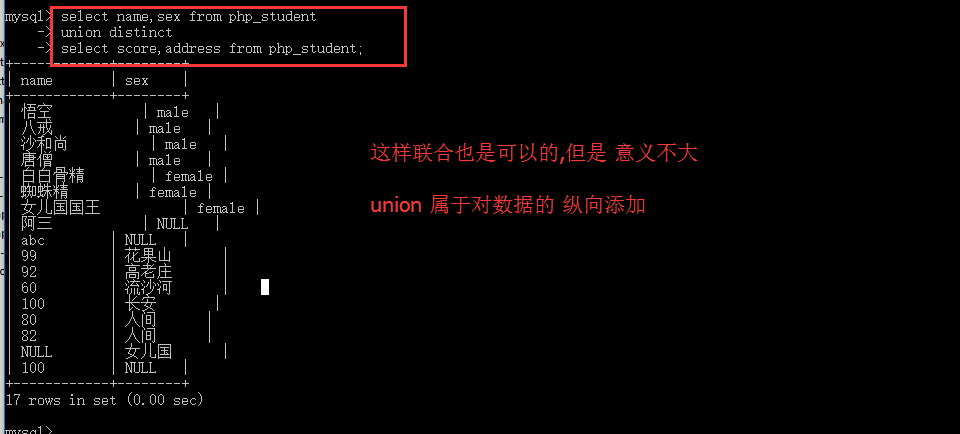
…;

选项就2个 一个是 all

另外一个 distinct 是默认选项

两个语句的字段列表中字段顺序、数目必须一致





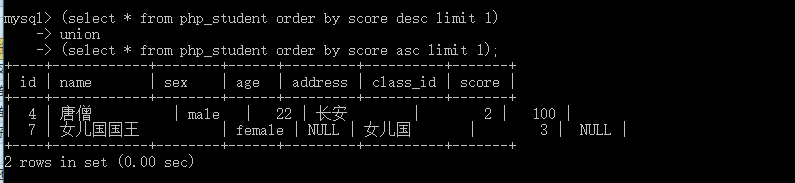
## union选项 all/distinct

all: 不去重

distinct: 去重

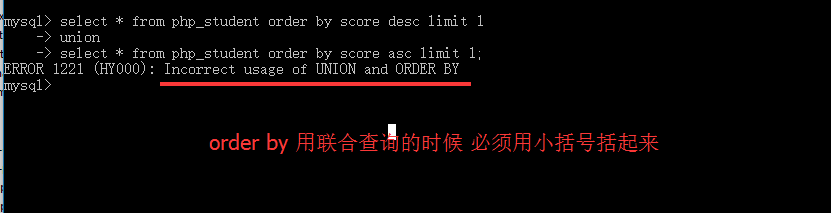
## union应用

1. 如果我们筛选数据的条件，在同一个select语句中存在逻辑上冲突，或者很难再同一个逻辑上实现，这个时候，需要把我们的需求拆分成多个逻辑，也就是在多个select语句上分别实现，然后再将结果拼接到一起就行了！



注意事项:

联合插入 如果有 order by 子句 最好用小括号括起来.



1. 如果一个网站的某张表的数据量特别的大，往往会导致查询效率特别低，此时，可以采取某种方式或规律对表进行“切割”，此时，每张表的结构还是一样的，只是存放的数据不一样！

比如：手机号码！

以河北为例:

保定

石家庄

廊坊

邯郸

# 第24讲 连接查询（交叉连接）

## 连接查询的概念

* 连接查询：通过指定的条件（共有字段）将多张数据表组织在一起，然后进行查询。共有字段是连接查询的基础。
* 根据查询的目的不一样，分为交叉连接、内连接、外连接。

## 交叉连接

交叉连接：cross join



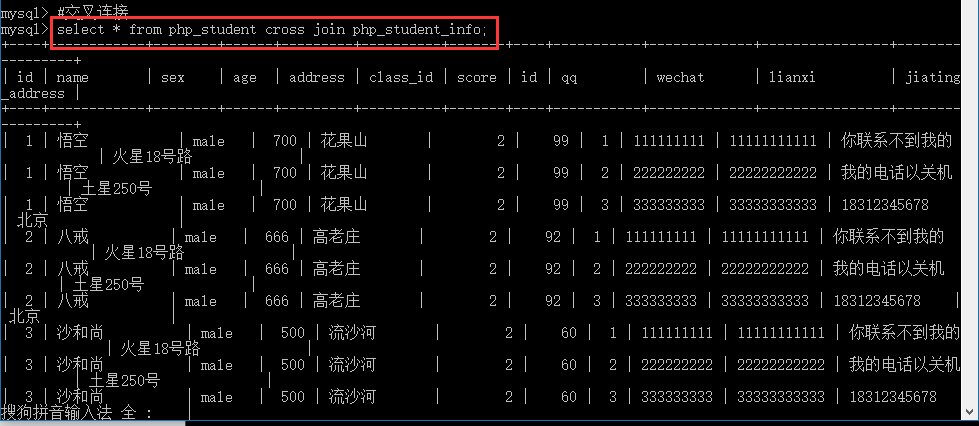
交叉连接就是 A的数据 和B的数据中的每一条数据进行连接

select \*|字段列表 from

表名 cross join 表名;

返回的结果的条数是 表名的记录条数\*表名的记录条数;

关键字是 cross join



交叉连接会产生很多无用的数据,所以交叉连接意义不大.

# 第25讲 连接查询（内连接）

## 含义

内连接的关键字是 inner join

内连接就分为左表和右表，出现在join关键字左边的就是左表，出现在join关键字右边的就是右表！

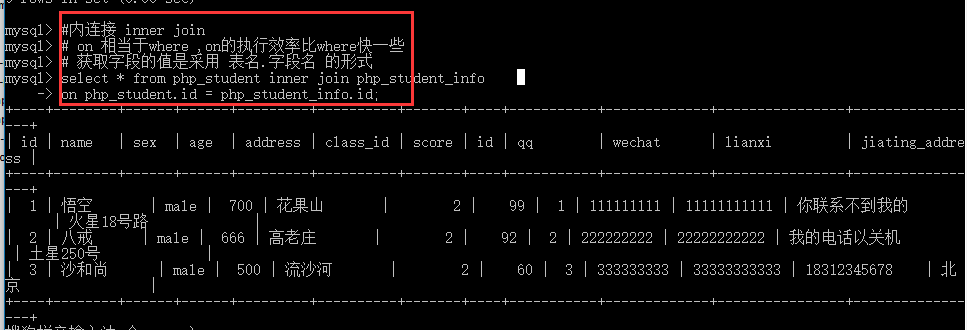
内连接首先做的还是笛卡尔积！数据在左表存在，同时在右表有对应的匹配的结果才会保存，如果没有匹配上，数据就没有意义，也就不会被保存！

## 语法规则

select \*|字段列表 from

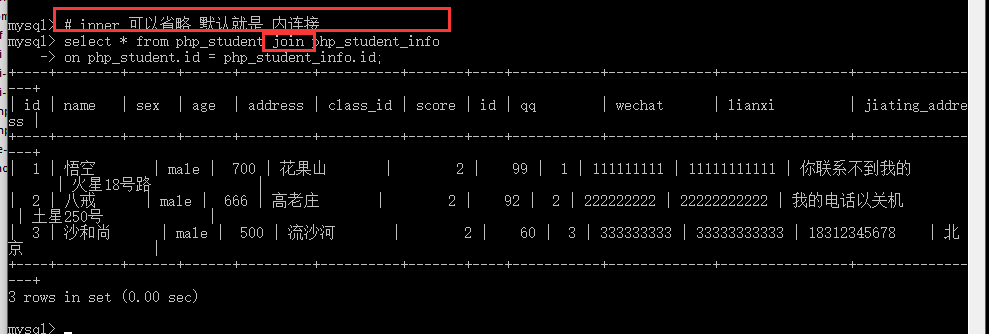
表 inner join 表 inner join 表 …

on 条件 ;



## 注意事项

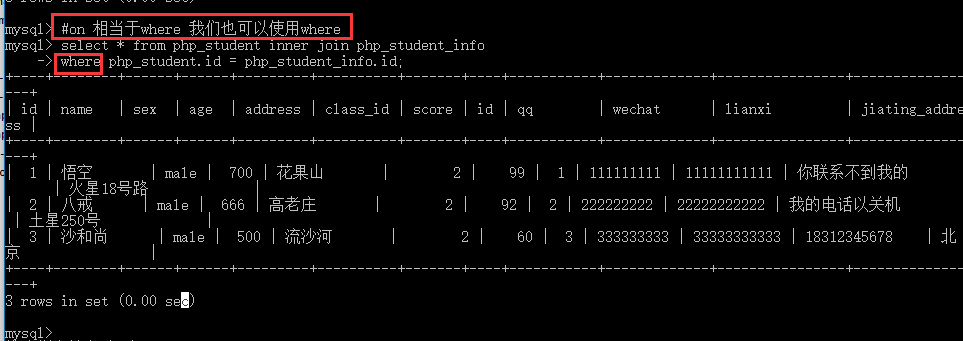
* Inner可以省略,只写join的时候,系统默认是就是内连接



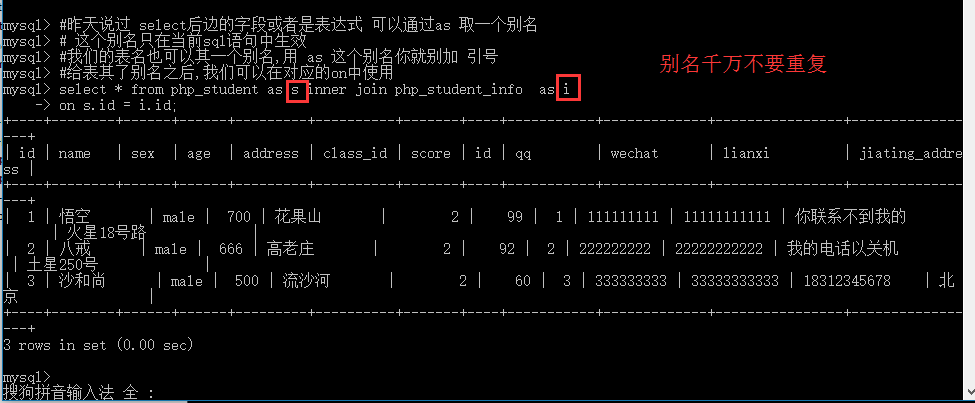
* on连接条件，不能省略。省略连接条件，返回笛卡尔积结果集。

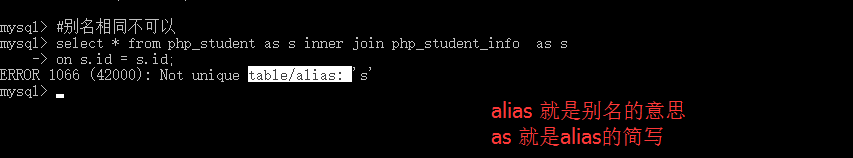


* on连接条件，可以使用where替换。on执行效率更高

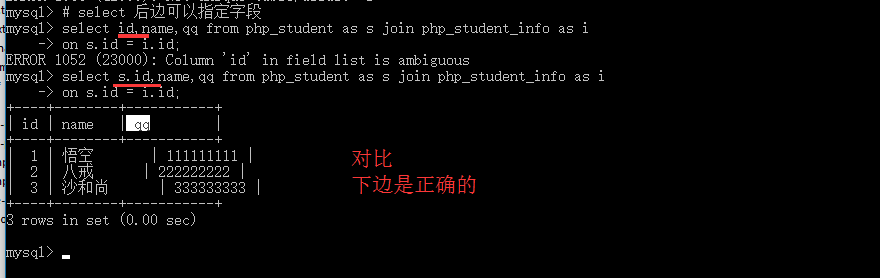


可以给表通过 as 其别名





如果我们再获取字段信息的时候,如果2个表中字段名有重复的,必须以 表名.字段 的形式来获取指定的数据



# 第26讲 连接查询（外连接）

外连接 分两种,左外连接和 右外连接

join

## 左外连接

左表为主,左表中无法连接的记录同样返回，右表中对应字段使用null填充。

select \*|字段列表 from

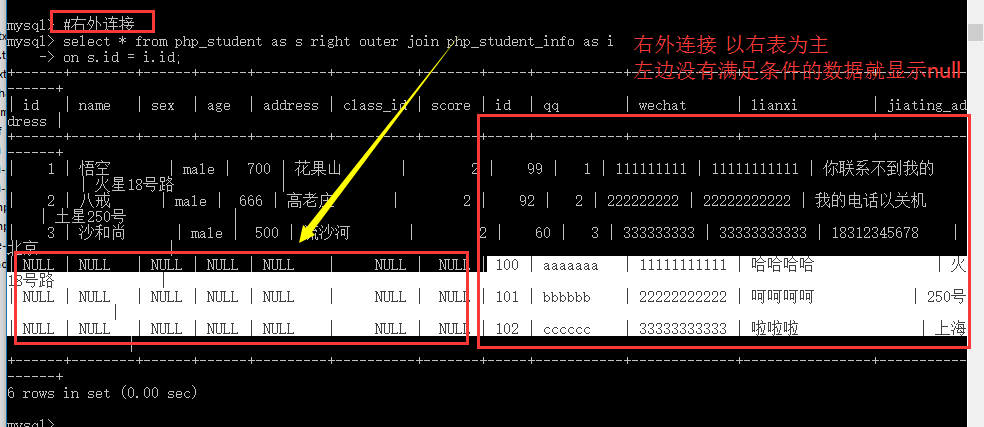
表 left outer join 表

on 条件;

## 

## 右外连接

右表为主,右表中无法连接的记录同样返回，左表中对应字段使用null填充。



## 注意事项

* 此时只能使用on不能使用where,为了我们很好的使用内连接和外连接,都使用on就OK了
* 左外连接一般叫做左连接,右外连接一般叫做右连接
* 左外连接和右外连接都可以省略掉outer

# 第27讲 子查询（标量子查询）

select \*|字段 from 表名

1\*(2+3)

我想查询班级中分数最高的人.

select max(score),name from php\_student; -- 错误!!!

明确第一个问题: max() 聚合函数 和分组配合使用. 默认没有分组默认是所有数据就是一组.

max 只会返回给我们一个分数的最大值

name 字段对应的值就是分组的第一条数据的 name值

select max(score) from php\_student;

select \* from php\_student where score = (select max(score) from php\_student);

## 子查询概念

* 一个查询的结果，作为另一个查询的条件，这个查询称之为子查询。（subquery）。相应的使用该条件的查询，称之为主查询。
* 子查询必须要放在小括号内()。子查询一般都是SELECT语句



## 子查询分类

标量子查询：子查询返回单一值

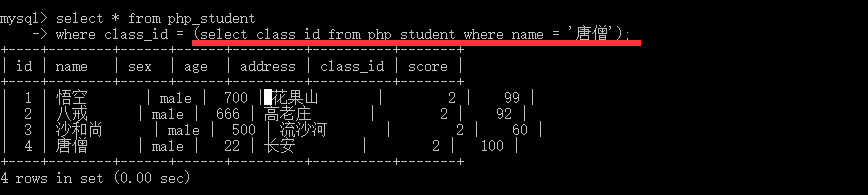
列子查询：子查询返回一列

行子查询：子查询返回一行

表子查询：子查询返回多行多列

## 标量子查询

* 单一的字段，在一定限定条件。

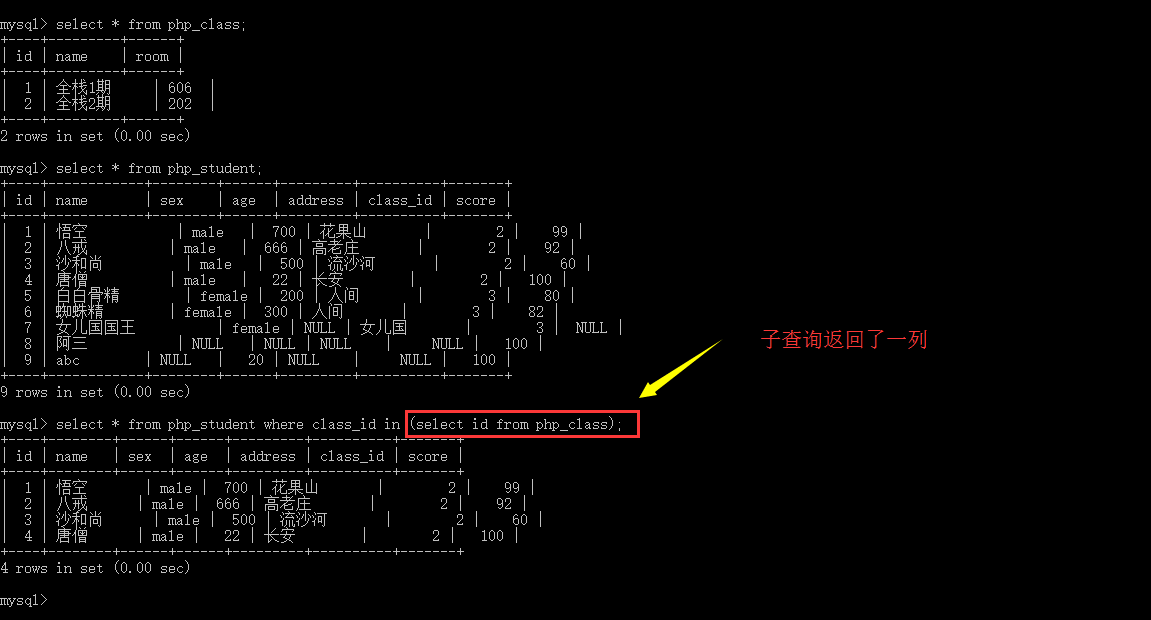


* 聚合函数（min,max,avg）



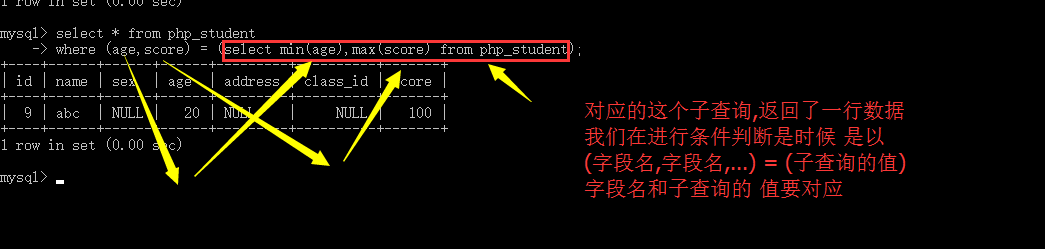
## 列子查询

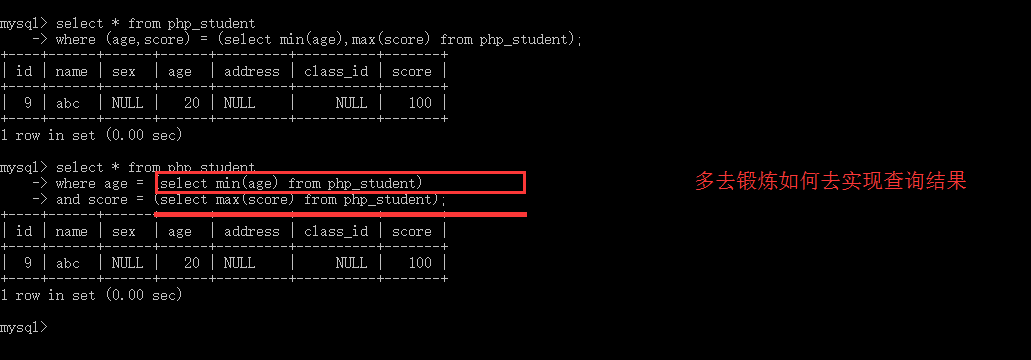
列子查询的结果往往就是一列数据的集合，所以，通常就是配合in或not in等集合运算符来使用！



## 行子查询

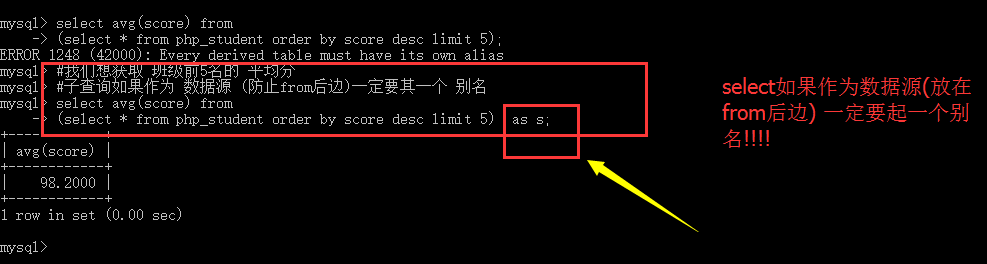
* 主查询的条件中的字段通常放在（）中。
* 主查询条件中字段顺序与子查询字段列表中字段顺序一致。





## 表子查询

返回结果是多行多列的子查询叫表子查询！



## exists子查询

exists主要是用来做判断，返回的是一个布尔值：true或者false

判断的依据：

如果子查询可以返回数据，则认为exists表达式就返回真，否则就返回假！

# 第28讲 范式

## 概念

范式就是我们设计表的基本规范！Normal Format

### 作用

通过合理的数据存储，从而使得数据的冗余度最小化以及运行效率的最大化！

### 分层

根据不同的需求标准，一层一层的严格递进，一层比一层严格，理论上来说，范式一共有6层！

* 比如：第一范式，第二范式……
* 要想满足第二范式，首先要满足第一范式，依次类推！但是，后面的范式实在是太严格了，很难达到，所以在数据库中，只引入三层范式！
* 满足三层范式的数据库就是设计比较合理的数据库！

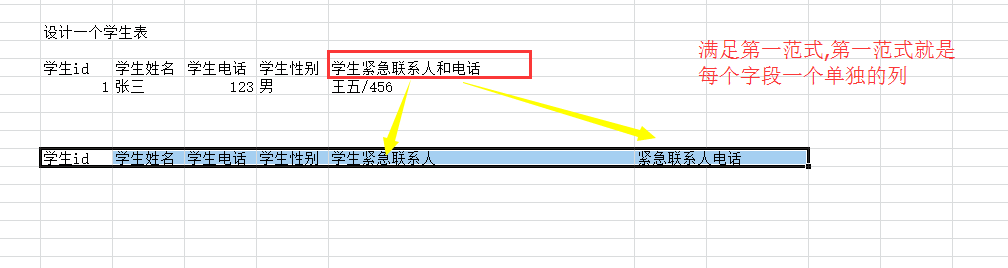
## 第一范式

### 概念

是最容易满足的，要求把各种数据设计成一个一个的单独的字段，不能再分割！

第一范式也叫满足“原子性”

例如: 性别和年龄设计在一个字段就不满足第一范式



## 第二范式

### 概念

就是在满足第一范式的基础上再满足以下的两个条件：

* 表中的每一行都具有唯一可区分的特性（就是不存在完全相同的记录）
* 不能有部分依赖

我们针对于第二范式的第一条规定,我们就是 定义一个主键!!!!!!!

### 什么叫依赖？

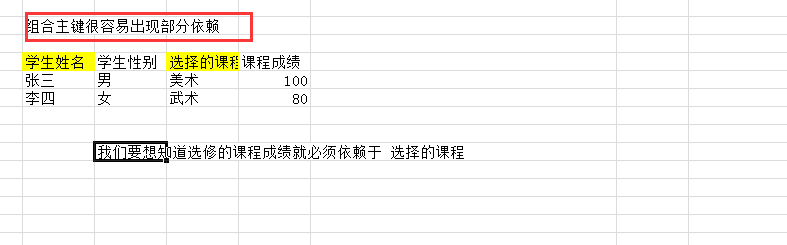
如果确定表中的某个字段A，就一定能确定该表的另一个字段B，那么我们就说B依赖于A

例如: 所有的非主键字段都依赖于主键

### 什么叫做部分依赖？

假如一个表的组合主键是（A，B），其他字段都应该依赖于（A，B），但是如果此时有一个字段C，它只依赖于A，也就是说，只要A确定了，C也就确定了，这种情况就叫作部分依赖

例如: 想要知道学生的选修分数,必须先知道选修的课程. 选修的学分就部分依赖于课程和学生



针对于部分依赖,我们需要 再单独设计一张表

## 第三范式

### 概念

在满足第二范式的基础之上消除传递依赖

### 什么是传递依赖？

如果一个表中有某个字段不直接依赖于主键，而是依赖于其他的字段，就叫作传递依赖！

* 假如主键是A，B依赖于A，而C又依赖于B！
* 注意：并不是说C字段不依赖于主键A，而是C只能等B确定后才依赖于A！
* 传递肯定是至少三个字段

例如: 想要了解一个学生他的院系电话和院系地址,这个时候需要先确定院系

学生id 学生姓名 学生院系 院系电话 院系地址

1 张三 法学系 123 顺义区京顺路99号

2 李四 法学系 123 顺义区京顺路99号

这么设计表是存在问题的,什么问题呢? 就是当 院系地址发生变化的时候,对所有的数据都有影响

解决传递依赖的方法还是 重新设计表

学生一个表 学生id, 学生姓名 ,学生院系id

院系表 院系id,院系电话,院系地址

部分依赖 和 依赖传递的关系:

依赖传递一定是部分依赖,但是 部分依赖不一定是传递依赖;

大家设计的表多了,慢慢就会形成规范了.

# 第29讲 数据的备份和还原

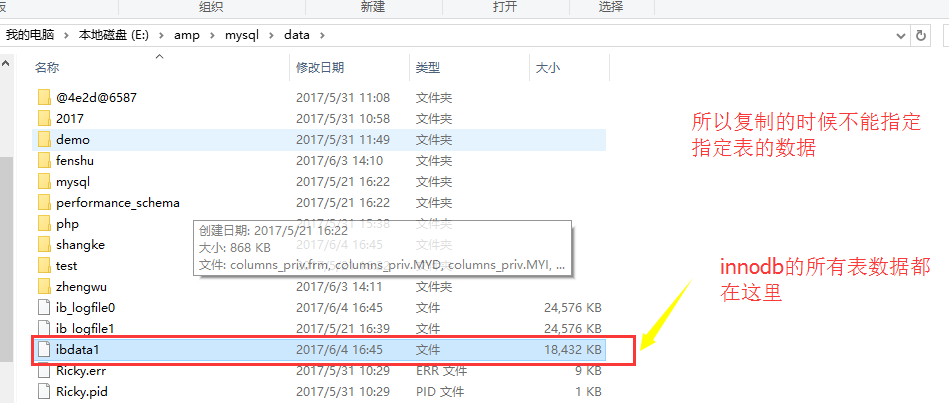
备份：就是将已有的内容复制一份，存放到其他的服务器上

## 文本备份

根据存储引擎的不同而不同.

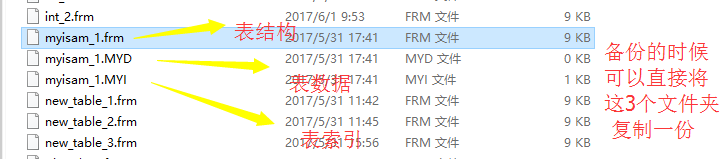
innodb

在对应的数据库的文件夹中 有一个表结构 表的数据在 ibdata



myisam

在对应 数据库的文件名文件夹下创建三个文件



备份的时候我们把 表结构,表数据,表索引都复制了一份.复制到另外一台服务器上

## 数据备份

只备份一张表的数据的部分

备份数据要用到sql语句

select \*|字段列表 into outfile 路径(绝对路径)

from 表名;

注意: 路径一定要用引号引起来,而且路径是 反斜杠 / (linux的斜杠)

## 

恢复数据

load data infile 路径 into table 表名(字段列表);

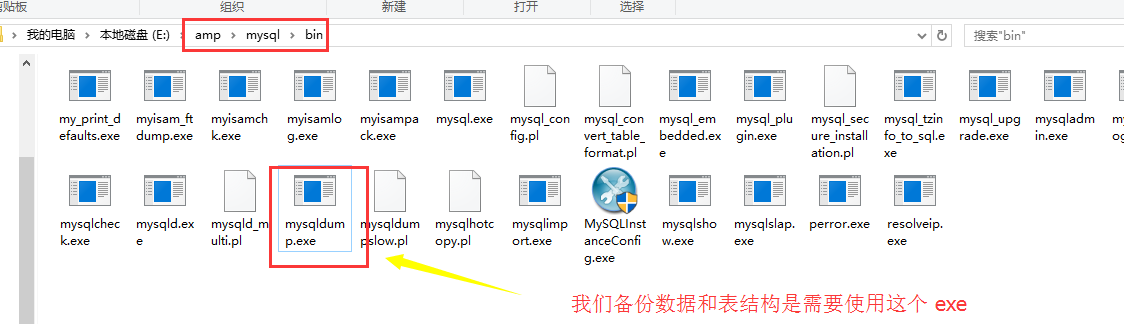
## 

## sql备份

将表的结构和数据通过sql语句的形式进行备份！

备份需要在cmd的环境中备份,cmd的环境变量是 mysqldump,

这个mysqldump 在对应的mysql/bin文件夹中



首先需要通过cmd到指定的 mysqldump这里

### 备份语法

mysqldump –h –P –u –p 数据库名 表名 > 路径

注意: 数据库名后边没有表名,表示整个库都备份

路径不要加 引号,也是用反斜杠

后缀名为 .sql

### 

### 还原语法

mysql –u –p 数据库名 < 路径

